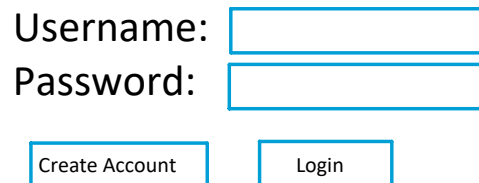


# Project Front end

Thursday, July 25, 2024 2:31 AM

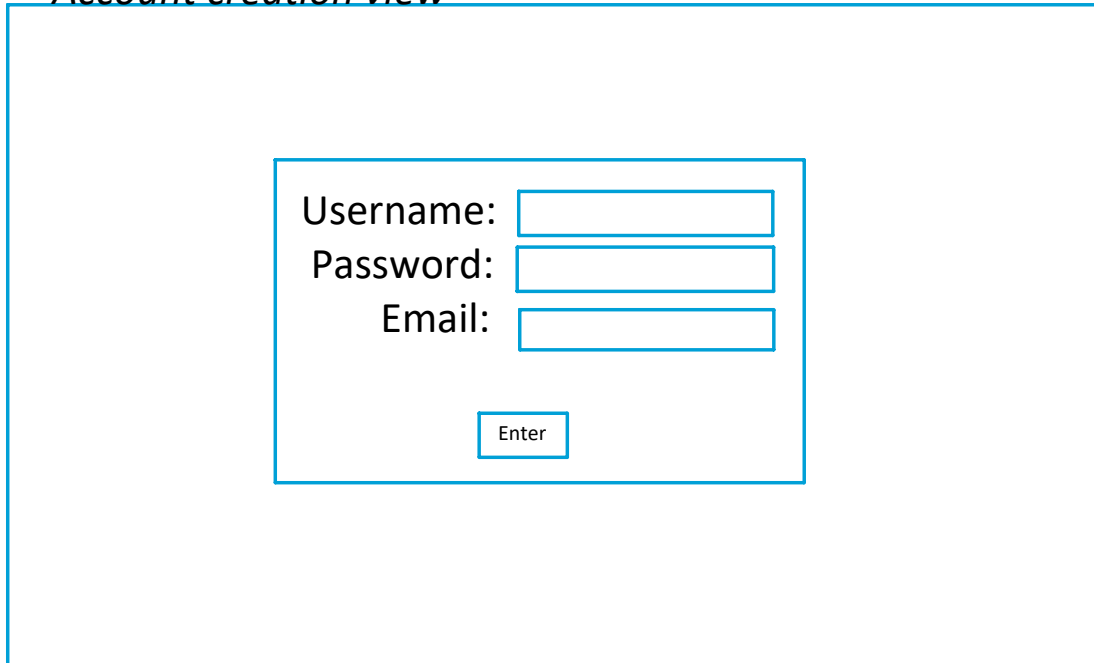
## *Login View*



A diagram of a login form. It consists of a light gray rectangular container. Inside the container, at the top left, is the text 'Username:' followed by a white rectangular input field. Below this is the text 'Password:' followed by another white rectangular input field. At the bottom of the container, there are two white rectangular buttons. The left button is labeled 'Create Account' and the right button is labeled 'Login'.

- User should input their unique login credentials and hit login, if their credentials don't match what's in the db we give a simple popup saying so and stay on this view
- If they don't have credentials the create account button takes them to the view where they create their account and input data into the db.
- If they do input valid credentials then we send them to the **account** view

### *Account creation view*




A diagram of an account creation form. It consists of a large outer rectangle containing a smaller inner rectangle. Inside the inner rectangle, there are three labels with corresponding input fields: 'Username:' followed by a text box, 'Password:' followed by a text box, and 'Email:' followed by a text box. Below these fields, centered, is a button labeled 'Enter'.

- The user would input credentials, if they are unique in the database they should be accepted and recorded and we can then send them to the **Account** view
- If the credentials aren't unique then the database should reject the insert event and we throw a pop up telling them to try again ie. "email already in use" or "Username already in use"

## Account view

{'username'}

Friends

 Settings

Logout

Create Character

{'Character 1'}

{'level','class','server'}

DELETE

{'Character 2'}

{'level','class','server'}

DELETE

{.....}

{'level','class','server'}

DELETE

{....}

{'level','class','server'}

DELETE

< Leaderboard >

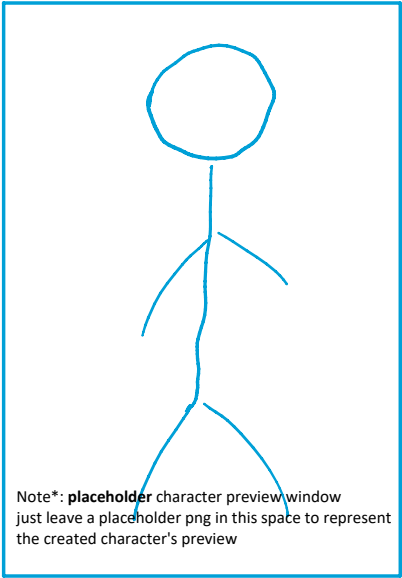








Rank	fame	name
1.	{'fame'}	{'name'}
2.	{'fame'}	{'name'}
3.	{'fame'}	{'name'}
4.	{'fame'}	{'name'}
5.	{'fame'}	{'name'}
6.	{'fame'}	{'name'}
7.	{'fame'}	{'name'}
8.	{'fame'}	{'name'}
9.	{'fame'}	{'name'}
10.	{'fame'}	{'name'}

Verified [ ]

Delete account

- Friends should take them to their friends view
- The settings cog should open up a little widget that has a verified checkbox that will update the accounts status to verified to simulate email verification.
- The delete account button should give a quick are you sure before deleting the user's account off the db and logging them out.
- Logout takes the user back to the login page
- The character select window on the left displays all characters owned by the account on the db.
- A user can delete the specified character using the delete button associated with that character.
- The create character button takes the user to the character creation view.
- Clicking on a characters name will bring the user to that specific character's view.
- The leaderboard shows the global and local leaderboards, you can cycle between the global and the various regional "local" leaderboards using the arrow buttons on the left and right.

## Character creation view

Name:	<input type="text"/>	
Class:	<input type="text"/>	
Race:	<input type="text"/>	
Age:	18  80	
Height:	120  240 cm	<p>Note*: <b>placeholder</b> character preview window just leave a placeholder png in this space to represent the created character's preview</p>
Weight:	30  200 kg	
Str:	1  10	
Dex:	1  10	
Int:	1  10	
Cha:	1  10	
Luck:	1  10	
<input type="button" value="Create Character"/>		

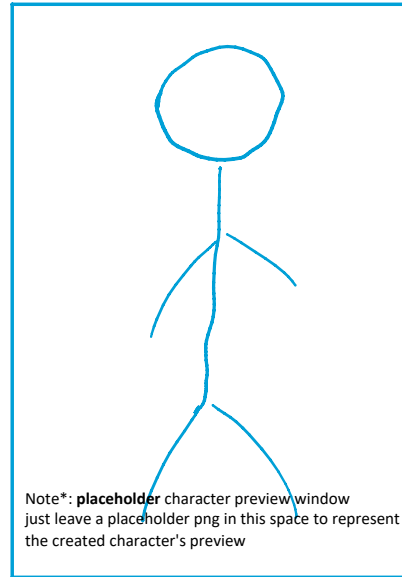
- The user will fill in the name field
- Class and race will be drop down menus with choices of {'mage', 'ranger', 'thief', 'pirate', 'warrior'} and {'elf', 'human', 'dwarf', 'halfling'} respectively.
- Age, height, weight, str, dex, int, cha, luck are all sliders.
- The stickman on the right is the character preview that we won't be implementing because it has nothing to do with sql or databases and we don't have art for it so just put a png of a stickman or something there to represent it.
- When the user hits the create character button, it should insert the character into the characters table and take the user back to the **account view**.
- Insertion here should be impossible to fail since all the variables are controlled with defaults and name doesn't need to be unique.

## Character view

Name:            {'name'}  
Level:           {'Level'}  
Class:           {'class'}  
Race:            {'race'}  
Str:             {'strength'}  
Dex:             {'dexterity'}  
Int:             {'intelligence'}  
Cha:             {'charisma'}  
Luck:            {'luck'}  
Money:           {'money'}

Inventory

Shared Inventory



Note\*: **placeholder** character preview window  
just leave a placeholder png in this space to represent  
the created character's preview

- In this view the user can view their created character and access their inventory
- Inventory takes the user to the inventory view owned by this character
- Shared inventory takes the user to the shared inventory view owned by this account
- The stats in this table should be affected by the equipment in the equipped table on the db, when equipment is added to equipped we add their stats and when they're removed we update the stats accordingly

Inventory View

Filter by/search:

Items

(\*debug shop)

Selected	Slot#	Name	stacks	equipped
[ ]	1.	{'item name'}	{'Stacks'}	[ ]
[ ]	2.	{'item name'}	{'Stacks'}	[ ]
[ ]	3.	{'item name'}	{'Stacks'}	[ ]
[ ]	4.	{'item name'}	{'Stacks'}	[ ]
[ ]	5.	{'item name'}	{'Stacks'}	[ ]

remove

Move to shared

- The filter/search box is used to query the inventory to only show specific items, probably implement it with regex searching for patterns in the item names or whether it is a consumable or equipment or resource.
  - Each of the headings in the item table should be able to sort the table by quickly querying the db and reloading the widget.
  - Equipped should be a toggleable box which looks at the equipped table in the db, if it is checked then the equipped table adds the relation, if its unchecked it removes the relation, if its not possible to equip the item the box should be greyed out. Ie. The user already has that type of equipment equipped or the type of item isn't equipment
  - The selected boxes should let the user select multiple things from their inventory, hitting remove with items selected will remove them from their inventory, hitting move to shared will attempt to move those items to the shared inventory
  - Move to shared will need to check the number of InvSlots on the account and update based on how many items we want to move. If the InvSlots attribute hits 0 we give a popup saying not enough inventory space instead of proceeding with the insert.
- Items will take the user to the items view which will be out standin for the shop, and let us demonstrate adding items to the inventory.

## Shared Inventory View

Filter by/search:

Selected	Slot#	Name	stacks
<input type="checkbox"/>	1.	{'item name'}	{'Stacks'}
<input type="checkbox"/>	2.	{'item name'}	{'Stacks'}
<input type="checkbox"/>	3.	{'item name'}	{'Stacks'}
<input type="checkbox"/>	4.	{'item name'}	{'Stacks'}
<input type="checkbox"/>	5.	{'item name'}	{'Stacks'}

remove

Move to character

- Literally the same as the inventory view but we track which character ID sent us to this view and use that as a target to transfer items to that character from the shared inventory, we also don't have an item shop or equippable to worry about here.
- The filter/search box is used to query the inventory to only show specific items, probably implement it with regex searching for patterns in the item names or whether it is a consumable or equipment or resource.
- The selected boxes should let the user select multiple things from their inventory, hitting remove with items selected will remove them from their inventory, hitting move to character will attempt to move those items to the characters inventory
- Move to cahracter will need to check the number of InvSlots on the character and update based on how many items we want to move. If the InvSlots attribute hits 0 we give a popup saying not enough inventory space instead of proceeding with the insert.

## Items View

Filter by/search:

Selected	Name	stacks	type	Description
[ ]	1. {'item name'}		{ 'type' }	{ 'description' }
[ ]	2. {'item name'}			
[ ]	3. {'item name'}			
[ ]	4. {'item name'}			
[ ]	5. {'item name'}			

Add to character


- Like the shared inventory view this view will need to remember which character brought us here.
- The filter/search box is used to query the items to only show specific items, probably implement it with regex searching for patterns in the item names or whether it is a consumable or equipment or resource.
- Each of the headings in the item table should be able to sort the table by quickly querying the db and reloading the widget.
- The stacks column will be filled with drop down menus that will let the user select a number between 1 and the item's max stacks if it has max stacks.
- The selected boxes should let the user select multiple things from the list, hitting add to character will attempt to add those items to the shared inventory.
- Add to character will need to check the number of InvSlots on the character and update based on how many items we want to move. If the InvSlots attribute hits 0 we give a popup saying not enough inventory space instead of proceeding with the insert.
- Hovering over the items description should cause a tooltip to appear containing the full description of the item.



### Friend view

{'username'}

Home

 Settings

Logout

Add friend

Search:

{'username1'}	DELETE
{'username2'}	DELETE
{.....}	DELETE
{....}	DELETE

Verified

[ ]

Delete account

- The search field lets you query the accounts table by username and populates a tooltip like popup, think like the ui on github when you add people to repos by searching their github name.
- The table on the left populates with all the current friends to the account
- The delete button lets you unfriend and delete that row from the friends table.
- Add friend just confirms the insert into friends table using the username selected by the search.
- The home button takes us back to the accounts view, everything else is the same as the accounts view.